Alternating CFG の拡張について(計算機科学の理論とその応用)

AUTHOR(S):

守屋, 悦朗; Otto, Friedrich; Messerschmidt, Hartmut

CITATION:

守屋, 悦朗 ...[et al]. Alternating CFG の拡張について(計算機科学の理論とその応用). 数理解析研究所講究録 2007, 1554: 9-15

ISSUE DATE:

2007-05

URL:

http://hdl.handle.net/2433/80978

数理解析研究所講究録
第1554巻 2007年 9-15

Alternating CFG の拡張について

(On Extensions of Alternating Context-Free Grammars)

早稲田大学・教育学部　守屋 悦朗 (Etsuro Moriya)
School of Education, Waseda University

Friedrich Otto　&　Hartmut Messerschmidt
Fachbereich Mathematik/Informatik, Universität Kassel, Germany

Abstract

交代性 CFG (ACFG) は，文脈自由文法 (CFG) の非終端記号に alternation を入れることにより [8] で導入された．[9] では，CFG に (alternation を含む) 状態を入れることにより状態交代性 CFG (sACFG) を導入し，交代性プッシュダウンオートマトン (APDA) の特徴付けを初めて与えた．また，[10] では，スタック記号に alternation を入れた PDA と，ACFG に (alternation の無い) 状態を導入した EACFG について考察し，ACFG と sACFG の特徴付けを与えた．本ノートでは，ACFG や sACFG の拡張を導入し，その生成能力について考察する．

The first attempt to introduce 'alternation' into the CFG to define the ACFG was proposed in [8]. Another alternating CFG called the state-alternating CFG (sACFG) was introduced in [9], where characterizations of the alternating pushdown automaton (APDA) by means of ACFGs were established. After that, further new variants of alternating CFG and PDA was introduced to obtain characterizations of ACFG and sACFG. In this note, we review the results obtained so far. Also, we introduce various extensions of alternating grammars and investigate relationships among them, a partial results of which has been announced elsewhere.

1 Introduction

交代性文脈自由文法 (ACFG) は交代性プッシュダウンオートマトン (APDA) を特徴付ける目的で [8] において導入された．ACFG (*alternating context-free grammar*) は5項組 $G = (V, U, \Sigma, P, S)$ で表わされるが，ここで，V は非終端記号アルファベット，$U \subseteq V$ は全称型 (*universal*) 非終端記号の集合，$V \setminus U$ は存在型 (*existential*) 非終端記号の集合，Σ は終端記号アルファベット，$S \in V$ は出発記号，P は文脈自由プロダクションの有限集合である．

G における導出の過程では，存在型非終端記号は通常の CFG における非終端記号と同様に書き換えが行なわれるが，全称型の非終端記号には，その記号を左辺に持つプロダクションすべてを同時に適用する (その結果，複数個の文形式が同時に生成される)．こうして，導出は文形式の1次元的連なりではなく，文形式をラベルとして付けられた木となる．終端語 w が G において生成されたとみなされるのは，そのような有限の木 T で次の条件を満たすものが存在する場合である：

(i) T の根には出発記号 S がラベル付けされている．
(ii) T のどの葉にも w がラベル付けされている．

以下では，文法のクラスを $\mathcal{G}$ とするとき，$\mathcal{G}$ の文法によって生成される言語のクラスを $\mathcal{L}(\mathcal{G})$ で表す．また，ε-プロダクションを持たない $\mathcal{G}$ の文法によって生成される言語のクラスを $\mathcal{L}(\varepsilon\text{-free-}\mathcal{G})$ で表す．さらに，最左導出に限定した文法によって生成される言語のクラスを $\mathcal{L}_{\mathsf{lm}}(\mathcal{G})$ および $\mathcal{L}_{\mathsf{lm}}(\varepsilon\text{-free-}\mathcal{G})$ で表す．

[8] では APDA の特徴付けは得られなかったが，その後，ACFG に関する興味深い結果がいくつか証明されている．例えば，[3] では，ACFG 言語のサブクラスと計算量クラスの間の密接な関係

$$\mathsf{P} = \mathsf{LOG}(\mathcal{L}(\text{linear-ACFG})), \quad \mathsf{PSPACE} = \mathsf{LOG}(\mathcal{L}_{\mathsf{lm}}(\varepsilon\text{-free-ACFG}))$$

が示されている．ここで，$\mathsf{LOG}(\mathcal{L})$ は言語のクラス $\mathcal{L}$ の対数領域還元の下での閉包を表し，linear-ACFG は線形プロダクションのみを持つ ACFG を表す．

また，[4] では，$\mathcal{L}(\mathsf{APDA}) = \mathcal{L}(\text{linear-erasing-ACFG})$ ということを証明することにより，不完全ながら $\mathcal{L}(\mathsf{APDA})$ の文法的特徴付けが与えられた．ここで，ACFG G が線形消去型 (*linear erasing*) であるとは，ある定数 c が存在して，G において生成される長さ n のどの語も長さが高々 $c \cdot n$ である文形式しかラベルとして含んでいないような導出木を持つことである．しかし，[4] で用いられている ACFG では，終端語 w に終止符 (*endmarker*) \$ を付けた $\$w\$$ という形の語だけを考えることにより導出のコントロールを行なっているので真正の ACFG ではない．

著者らは [9] において，[5] により導入された状態付き CFG (ECFG と略記する) の概念と alternation の概念とを結びつけることにより，状態付き ACFG (sACFG) を導入し，APDA の文法的特徴付けに成功した：

$$\mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG}) = \mathcal{L}(\mathsf{APDA}).$$

ただし，$\mathcal{L}(\mathsf{ACFG}) = \mathcal{L}(\mathsf{APDA})$ であるかどうかは未だに未解決である ($\mathcal{L}(\mathsf{ACFG}) \subseteq \mathcal{L}(\mathsf{sACFG})$ は証明されている)．

sACFG (*state-alternating context-free grammar*) では，ACFG がそうであるように非終端記号を全称型なものと存在型のものに分けるのではなく，状態を全称的なものと存在的なものに分けている．しかし，状態を全称型のものと存在型のものに分けずに，非終端記号だけを全称型のものと存在型のものに分けても生成能力が変わらないことが [9] で証明されている．すなわち，Fig.1 において

$$\mathcal{L}(\mathsf{EACFG}) = \mathcal{L}(\mathsf{sACFG}), \quad \mathcal{L}_{\mathsf{lm}}(\mathsf{EACFG}) = \mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG})$$

が成り立っている．

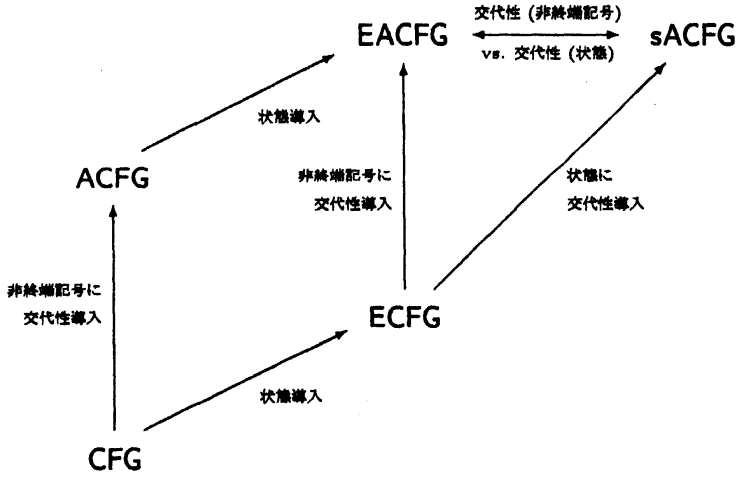

Figure 1: ACFG の間の関係

また，著者らは [10] において，APDA の状態を全称型と存在型に区別せず，スタック記号を全称型と存在型に区別するモデル stackAPDA を導入し，そのような APDA の受理能力は

オリジナルの APDA の受理能力と変わらないことを示した．同時に，stackAPDA のサブクラス $\mathsf{stackAPDA}_0$ (状態を持たない stackAPDA) を用いて $\mathcal{L}_{\mathsf{lm}}(\mathsf{ACFG})$ を特徴付けた．すなわち，Fig.2 において

$$\mathcal{L}(\mathsf{stackAPDA}) = \mathcal{L}(\mathsf{APDA}), \quad \mathcal{L}_{\mathsf{lm}}(\mathsf{ACFG}) = \mathcal{L}(\mathsf{stackAPDA}_0)$$

が成り立っている．

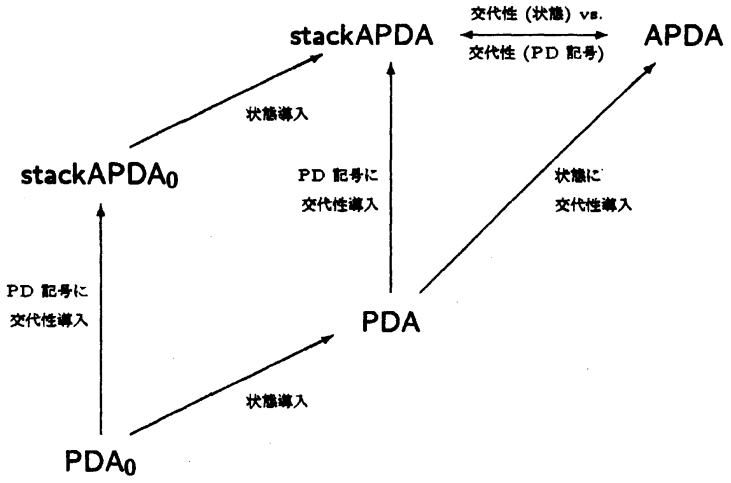

Figure 2: APDA の間の関係

最後に，最左導出については述べるまでもないが，制限最左導出 (*leftish derivation*) を次のように定義する．これは [5] において初めて導入された概念である．$(p, A) \to (q, \alpha)$ がプロダクションであり，β のいかなるプレフィックスにも状態 p の下で適用できるプロダクションが無いとき，直接導出 $(p, \beta A\gamma) \Rightarrow (q, \beta\alpha\gamma)$ は制限最左であるという．すなわち，β は非終端記号を含んでいてもよいが，状態 p の下ではそれを書き換えるプロダクションが存在せず，p の下でプロダクションが適用できる最左の非終端記号が A であるというのが制限最左の条件である．ACFG や ε-プロダクションを持たない ACFG によって制限最左導出の下で生成される言語のクラスを $\mathcal{L}_{\mathsf{lt}}(\mathsf{ACFG})$ および $\mathcal{L}_{\mathsf{lt}}(\varepsilon\text{-}\mathsf{free}\text{-}\mathsf{ACFG})$ で表す．

本稿では，ACFG の拡張について考察する．基礎になる文法として CFG より生成能力の高い文法や，それに状態を持たせた文法 (ECFG の拡張) 等を考え，状態を全称型と存在型に分けることや非終端記号を全称型や存在型に分けることにより生成能力がどのように変わるかや，最左導出や制限最左導出の効果について考察する．例えば，基礎文法として一般の 0 型文法を考えても，最左導出に制限すれば，状態を全称型と存在型に分けようと，非終端記号を全称型や存在型に分けようと生成能力は変わらないことが示される (定理 2.3)．このことは，基礎文法が CFG の場合には未解決の問題である．すなわち，[9] において $\mathcal{L}(\mathsf{ACFG}) \subseteq \mathcal{L}(\mathsf{sACFG})$ や $\mathcal{L}_{\mathsf{lm}}(\mathsf{ACFG}) \subseteq \mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG})$ は証明されているが，等号が成り立つか否かはわかっていない．

2 (s)ACFG をいかに拡張するか

交代性 (alternation) の概念を Chomsky 階層のそれぞれの文法クラスに導入することはごく自然なことである．この節では，Chomsky 階層における i 型の文法の交代性バージョンを考え，Chomsky 階層に属す本来の文法に対して成り立つ事実が交代性バージョンの文法ではどうであるかを考察する．

ACFGの拡張とも言える交代制の0型文法の定義から始めよう．0型文法の交代性バージョンを定義するに際して，その基底となる文法としてプロダクションが次の形であるようなものが先ず考えられる：

$$\alpha X \beta \to \alpha \gamma \beta \tag{2.1}$$

ここで、X は非終端記号，α, β は非終端記号からなる文字列，γ は終端記号と非終端記号からなる文字列である．

プロダクションの形 (2.1) はCFプロダクションの一般化としてごく自然なものであるから，この定義は最も自然な定義のように見える．(2.1) の形のCFプロダクションだけで任意の0型言語が生成できることも知られているとおりである．

さて，ACFGを定義したときと同様に，交代性の0型文法 (Atype0 で表す) の非終端記号を2種類に分割する．すなわち，$U \subseteq V$ を全称的非終端記号の集合とし，その補集合 $V \setminus U$ を存在的非終端記号の集合とする．こうして，交代性0型文法を5項組 $G = (V, U, \Sigma, P, S)$ で表す．ここで，プロダクションの形が違う (上述) ということを除けば，G を構成する各要素はACFGのそれとまったく同じものを表す．

さて，プロダクションの型 (全称型か存在型か) を定義するために，文字列の型を先に定義する．終端記号と非終端記号の混じった文字列において，最も左側に現れる非終端記号が全称型であるとき，その文字列は全称型であると定義する．そうでない場合 (最左の非終端記号が存在型である場合) は存在型であるという．これに基づき，プロダクションの場合，その左辺の文字列が全称型か存在型かでそのプロダクションの型を定義する．(2.1) の形のプロダクション $\alpha X \beta \to \alpha \gamma \beta$ において，X が全称型であるか存在型であるかに従ってプロダクションの型を定義する方法も考えられるが，この定義ではプロダクションが全称型であるか存在型であるかが一意的に定まらない．そこで，0型文法のプロダクションとして (2.1) を用いることはせず，次の形のプロダクションを持つ文法を基底文法として用いることにする：[1]

$$\alpha \to \eta \tag{2.2}$$

ここで，α は非終端記号だけからなる文字列であり，η は終端記号と非終端記号が混在する文字列である．(2.2) において $|\alpha| \leqq |\eta|$ が成り立っている場合，この文法は1型文法であるとか，文脈依存文法であるとか，単調文法であるといい，この文法のクラスを ACSG で表すことにする．

次に，与えられた0型文法に対する導出木をACFGに対するそれと同様に定義する．そのために，プロダクションが全称的に適用されるとはどういうことなのか (存在的に適用される場合も同様) を述べよう．n をある導出木の1つのノードとし，そこには文形式 $\beta\alpha\gamma$ がラベル付けされていたとしよう．もし，左辺が α であるプロダクションが全部で

$$\alpha \to \eta_1, \ \ldots, \ \alpha \to \eta_k$$

であったときには，これらのプロダクションすべてが同時並列的に文形式 $\beta\alpha\gamma$ の中の α に適用されなければならず，その結果，ノード n には k 個の子が生成され，それらにはラベル $\beta\eta_1\gamma, \ \ldots, \ \beta\eta_k\gamma$ が付けられる．もしさらに，β が終端記号でなければならないという制限を付ける場合，この生成 (導出) は最左であるという．もし，最左の部分文字列として，プロダクションの左辺であるようなものが複数あった場合には，そのなかの1つが非決定的 (nondeterministically) に選ばれるものとする．例えば，$\alpha' \to \eta'_1, \ \ldots, \ \alpha' \to \eta'_{k'}$ もまた P のプロダクションであり (左辺が α' であるプロダクションのすべて)，かつ α' が $\beta\alpha\gamma$ の部分文字列であり，$\alpha \neq \alpha'$ であった場合を考えよう．α と α' のどちらかが全称的である場合と，どちらも全称的である場合とがある．いずれの場合も α または α' のどちらかが非決定的に選ばれる．もし全称的な α

[1] ここでは述べないが，このような定義以外にも自然と思われる定義がいくつか考えられ，それらが一致することも証明できる．

(あるいは α') が選ばれた場合，左辺が α (あるいは α') のプロダクションすべてが同時並列的に適用されるというのが定義である．選ばれた α (あるいは α') が存在的であった場合，左辺が α (あるいは α') のプロダクションのなかの1つが非決定的に選ばれて適用される．

[9] と同様に，X 型の文法によって最左導出で生成される言語のクラスを $\mathcal{L}_{\mathsf{lm}}(\mathsf{X})$ で表すことにする．

最初に，Atype0 文法の最左導出に関する生成能力について考察する．以下の補題が証明できる．

Lemma 2.1. $\mathcal{L}_{\mathsf{lm}}(\mathsf{Atype0}) \subseteq \mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG})$.

Lemma 2.2. $\mathcal{L}_{\mathsf{lm}}(\mathsf{sAtype0}) \subseteq \mathcal{L}_{\mathsf{lm}}(\mathsf{Atype0})$.

補題 2.1 と 2.2，および $\mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG}) \subseteq \mathcal{L}_{\mathsf{lm}}(\mathsf{sAtype0})$ が成り立つという自明な事実より，次の定理が得られる：

Theorem 2.3. $\mathcal{L}_{\mathsf{lm}}(\mathsf{Atype0}) = \mathcal{L}_{\mathsf{lm}}(\mathsf{sAtype0}) = \mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG})$.

明らかに $\mathcal{L}_{\mathsf{lm}}(\mathsf{ACSG}) \subseteq \mathcal{L}_{\mathsf{lm}}(\mathsf{Atype0})$ が成り立つから，次の系も得られる：

Corollary 2.4. $\mathcal{L}_{\mathsf{lm}}(\mathsf{ACSG}) \subseteq \mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG}) = \mathcal{L}(\mathsf{APDA})$.

補題 2.2 の証明に依存することであるが，この時点では系 2.4 の逆の包含関係が成り立つかどうかはわからない．

さて，交代性LBA (alternating linear bounded automaton) のクラスを ALBA で表すことにする．$\mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG})$ と $\mathcal{L}(\mathsf{ACSG})$ の間の包含関係が知りたい．というのは，$\mathcal{L}(\mathsf{APDA}) = \mathcal{L}(\mathsf{ALBA})$ [2] および $\mathcal{L}(\mathsf{APDA}) = \mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG})$ [9] が成り立つことが知られているからである．

暫くの間，最左導出に制限しない導出 (以下，非最左導出と呼ぶ) のもとで ACSG について考えよう．次の補題が必要である．

Lemma 2.5. 任意の ALBA に対し，その受理状態がすべて存在的であるような等価な ALBA が存在する．

Lemma 2.6. M を ALBA とするとき，非最左導出の下で $L(M)\#_1\#_2\#_3$ を生成する ACSG G が存在する．ここで，$\#_1, \#_2, \#_3$ は M の入力アルファベットには属さない記号である．

さらに，次のややテクニカルな補題が必要である．

Lemma 2.7. $L \subseteq \Sigma^+$ を言語とし，$\#$ は Σ の元ではない記号とする．もし $L\#$ が ACSG G によって非最左導出の下で生成されるならば，L を非最左導出の下で生成する ACSG G' が存在する．

補題 2.6 と 2.7 から，次の補題が得られる：

Lemma 2.8. $\mathcal{L}(\mathsf{ALBA}) \subseteq \mathcal{L}(\mathsf{ACSG})$.

また，系 2.8 の逆も証明することができる：

Lemma 2.9. $\mathcal{L}(\mathsf{ACSG}) \subseteq \mathcal{L}(\mathsf{ALBA})$.

したがって，次の定理が証明された：

Theorem 2.10. $\mathcal{L}(\mathsf{ACSG}) = \mathcal{L}(\mathsf{ALBA})$.

Corollary 2.11. $\mathcal{L}(\mathsf{ACSG}) = \mathcal{L}(\mathsf{APDA}) = \mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG})$.

系 2.4 と 2.11 から，次の系も得られる：

Corollary 2.12. $\mathcal{L}_{\mathsf{lm}}(\mathsf{ACSG}) \subseteq \mathcal{L}(\mathsf{ACSG})$.

残された問題は，系 2.12 の逆の包含関係が成り立つかどうかということである．系 2.11 より，これは包含関係 $\mathcal{L}_{\mathsf{lm}}(\mathsf{sACFG}) \subseteq \mathcal{L}_{\mathsf{lm}}(\mathsf{ACSG})$ が成り立つかどうかと等価である．ε-free sACFGs に制限するならばこの包含関係は成り立つ：

Lemma 2.13. $\mathcal{L}_{\mathsf{lm}}(\varepsilon\text{-free sACFG}) \subseteq \mathcal{L}_{\mathsf{lm}}(\mathsf{ACSG})$.

最後に，逆の包含関係について少し考えてみよう．k を正整数とする．任意の $w \in L(G)$ に対し，G において w を生成する最左導出木が存在し，この導出木上のどの道においても ε-プロダクションの適用回数が $k \cdot |w|$ 回以下であるとき，sACFG G は *k-ε-bounded* であると定義する．G が k-ε-bounded であるような正整数 k が存在するとき，G は *ε-bounded* であるという．上記の補題より，次のことが成り立つと予想される：

Conjecture. $\mathcal{L}_{\mathsf{lm}}(\varepsilon\text{-bounded sACFG}) \subseteq \mathcal{L}_{\mathsf{lm}}(\mathsf{ACSG})$.

これまで述べた結果を以下に図示する．

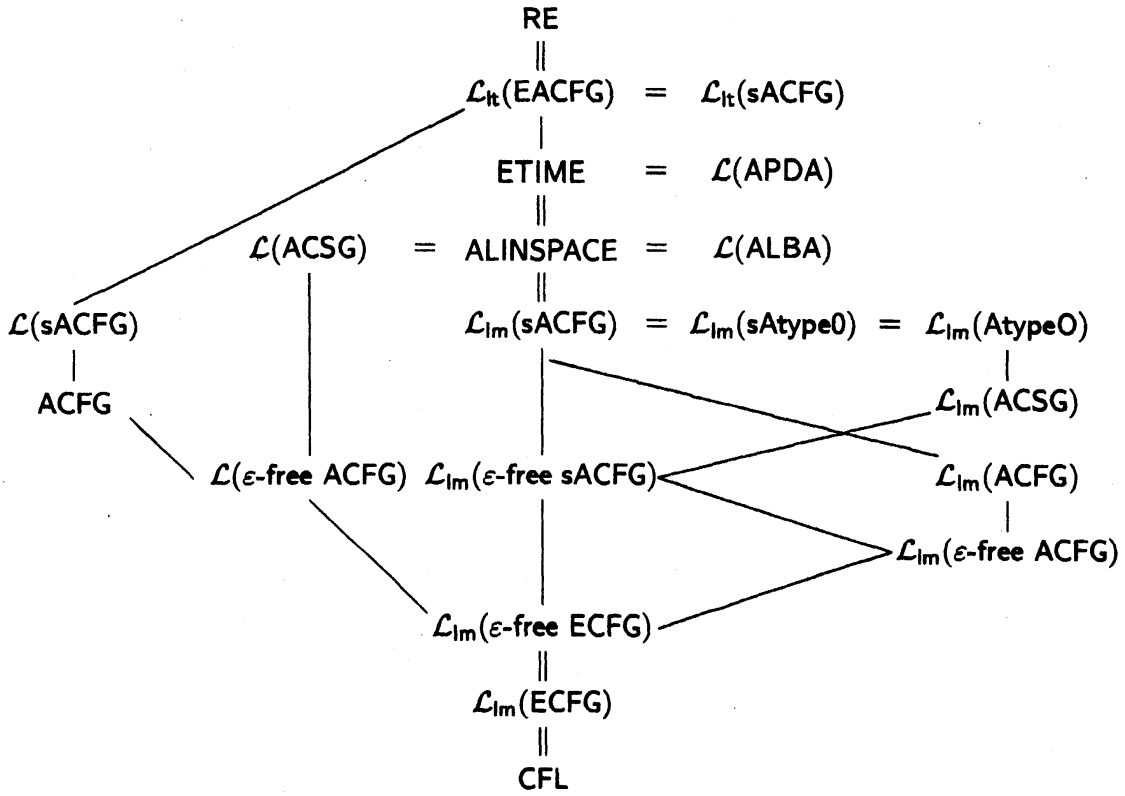

Figure 3: Inclusion relations among classes of alternating languages

Acknowledgement 筆頭著者は，この研究の一部に対し早稲田大学特定課題研究#2006B-073 の助成を受けたことを感謝する．

References

[1] G. Buntrock and F. Otto. Growing context-sensitive languages and Church-Rosser languages. *Information and Computation*, 141:1–36, 1998.

[2] A. K. Chandra, D. C. Kozen and L. J. Stockmeyer, Alternation, *J. ACM* **28**, 114–133, 1981.

[3] Z. Z. Chen and S. Toda. Grammatical characterizations of P and PSPACE. *The Transactions of the IEICE*, E 73:1540–1548, 1990.

[4] O. H. Ibarra, T. Jiang, and H. Wang. A characterization of exponential-time languages by alternating context-free grammars. *Theoretical Computer Science*, 99:301–313, 1992.

[5] T. Kasai. An infinite hierarchy between context-free and context-sensitive languages. *Journal of Computer and System Sciences*, 4:492–508, 1970.

[6] R. E. Ladner, R. J. Lipton, and L.J. Stockmeyer. Alternating pushdown automata. In *Proceedings of the 19th FOCS*, pages 92–106. IEEE Computer Society Press, 1978.

[7] M. Lange, Alternating contex-free languages and linear time μ-calculus with sequential composition, *Electric Notes in Theor. Comput. Sci.*, 2002.

[8] E. Moriya, A grammatical characterization of alternating pushdown automata, *Theor. Comput. Sci.* 67, 75–85, 1989.

[9] E. Moriya, D. Hofbauer, M. Huber and F. Otto, On state-alternating context-free grammars, *Theor. Comput. Sci.* 337, No.1–3, pp.183–216, June, 2005.

[10] E. Moriya and F. Otto, Two ways of introducing alternation into context-free grammars and pushdown automata, Oct., 2006, to appear in *IEICE Trsns. on Inform. Systems.*